CR-189384

FINAL
IN-82
OCIT
33947
P. 18

Final Report

August 31, 1994

Project Title:
An Incremental Database Access Method for Autonomous Interoperable Databases

Contract Number USRA 5555-09

Principal Investigator: *Nick Roussopoulos*
Co-Principal Investigator: *Timos Sellis*

Department of Computer Science
University of Maryland
College Park, MD 20742

Summary of Research

Commercially available database systems do not meet the information and processing needs of scientist who need to access multiple autonomous databases organized and maintained under a variety of DBMSs. With the accessibility of these databases via readily available computer networks, users will demand the capability to jointly manipulate data in different databases. The objective of this project is to illustrate the concept of incremental access to distributed and autonomously managed databases. An experimental database management system, ADMS±, which has been developed at the University of Maryland, in College Park, is used as the testbed of this research.

In this research, we investigated a number of design and performance issues of interoperable database management systems. The major results of our investigation were obtained in the areas of Client-Server Database architectures for heterogeneous DBMSs, incremental computation models, buffer management techniques, and query optimization. We finished a prototype of an advanced Client-Server workstation-based DBMS which allows access to multiple heterogeneous commercial DBMSs. Experiments and simulations were then run to compare its performance with the standard client-server architectures.

The focus of this research was on adaptive optimization methods of heterogeneous database systems. Adaptive buffer management accounts for the random and object oriented access methods for which no known characterization of the access patterns exist. Adaptive query optimization means that value distributions and selectivities, which play the most significant role in query plan evaluation, are continuously refined to reflect the actual values as opposed to static one that are computed offline.

Query feedback is a concept that was first introduced to the literature by our group. We employed query feedback for both adaptive buffer management and for computing value distributions and selectivities. For adaptive buffer management, we use the page faults of prior executions to achieve more "informed" management decisions. For the estimation of the distributions of the selectivities, we use curve-fitting techniques, such as "least squares" and "splines", for regressing on these values.

1. ViewCache: An Incremental Access Method for Database Access

Query processing in relational databases can be sped up by maintaining views as stored sets of pointers, called *ViewCaches*, pointing to those tuples in the underlying relations needed to materialize the views. A *ViewCache* is a stored collection of pointers pointing to records of underlying relations needed to materialize a view [Roussopoulos 1991]. In this paper, an *Incremental Access Method (IAM)* that amortizes the maintenance cost of ViewCaches over a long time period or indefinitely is developed and studied analytically and experimentally. Amortization is based on *deferred* and other update propagation strategies. A deferred update strategy allows a ViewCache to remain outdated until a query needs to selectively or exhaustively materialize the view. At that point, an incremental update of the ViewCache is performed. This paper defines a set of conditions under which incremental access to the ViewCache is cost-effective. The decision criteria are based on some dynamically maintained cost parameters which provide accurate information but require inexpensive bookkeeping.

The IAM capitalizes on the ViewCache storage organization for performing the update and the materialization of the ViewCaches in an interleaved mode using one--pass algorithms. Compared to the standard technique for supporting views that requires reexecution of the definition of the view, the IAM offers significant performance advantages. We will show that under favorable conditions, most of which depend on the size of the incremental update logs between consecutive accesses of the views, the incremental access method outperforms query modification. Performance gains are higher for multilevel ViewCaches because all the I/O and CPU for handling intermediate results is avoided.

The IAM allows for a number of update strategies to be implemented:

a) **Immediate update broadcast**

All affected ViewCaches are immediately updated. This strategy maintains minimal response time at the expense of system overhead. For some applications and some views, this may be desirable for instant response time on view retrieval.

b) **Periodic update broadcast**

All affected ViewCaches are periodically updated. This strategy can also be used to do updates during periods of low system load or at prespecified times.

c) **Event triggered update**

A ViewCache is updated when an event occurs. For example, a particular event can be triggered when the thresholds for the IAM cost-effectiveness on the incremental update logs have been reached (selective update propagation).

d) **Deferred update**

All ViewCaches stay outdated until a direct or indirect reference to them is made; at that time, the IAM for a ViewCache is invoked if it is cost-effective. This avoids any system overhead associated with immediate propagation of updates.

Strategies (b)-(d) implement the *deferred (lazy)* approach, that postpones the incremental update propagation either for a while (strategies (b) and (c)) or indefinitely (strategy (d)) unless a direct or indirect reference to a ViewCache is made.

The algorithms and the results obtained for the IAM are applicable to all strategies (a)-(d). Therefore, the strategies can be combined, allowing the user to specify different strategies for views depending on their importance, frequency of access, and other application dependent criteria. The main difference among these strategies is how they control the size of the incremental update logs. The trade-off is between system overhead and query processing time. The deferred strategies incur less or no overhead on the system at the cost of having to process bigger incremental update logs during the query processing.

In addition to the theoretical (worst case) analysis presented in [Roussopoulos 1991], experiments were run on our prototype runing on a Sun 3/280. These compared the reexecution indexed loop join algorithm with the corresponding incremental one. Both algorithms use a single B-tree as a secondary index. The experiments consisted of mixed complexity queries run several times against the database. Complete analysis can be found in [Roussopoulos 1991] for a variety of query/update loads drawn from the Wisconsin benchmark.

In [Roussopoulos et al 1993] we presented three more applications of the Incremental Access Methods including a Prolog record-at-a-time access method facilitated by VIEWCACHE and its IAM and an architecture for a multi-site federation of database client-servers. In the latter architecture, each client workstation not only submits database requests to the servers, but in addition, it runs a DBMS caching into its local disk data transmitted during query processing. The Client-Server architecture paradigm is the foundation for distributed and cooperative processing in several application areas. Such a paradigm is of paramount importance as the size and complexity of databases increase. Downloaded data is then on accessed locally, thus, saving data transmission over the network. Updates are done on the servers which act as the primary sites of the data stored in them.

Downloaded cached data is maintained incrementally, by transmitting, on demand, from the appropriate servers the updated records of the relevant data between two consecutive requests of each workstation. The main advantage of this architecture is that it distributes query processing and disk accessing on multiple workstations. and this save to it. The performance is tremendously increased because all cached data is accessed in parallel. Thus, this enhanced client-server architecture enjoys features of database machines with multiple disks. We have developed simulation packages for measuring the impact of this architecture on query throughput. The simulations show that, depending on the frequency of updates, speed-ups of up to linear in the number of participating workstations. It also shows that the architecture scales up very in a very satisfactory manner.

The incremental access methods described above provide the foundation for client-server database architectures described in the next sections.

1.1. Client-Server Database Architectures

The use of multiprocessor systems in databases and database machines have been thoroughly investigated over the past decade. These efforts tried to predominantly optimize the performance of large join operations by utilizing multiple disks and processors. Aside from their limited commercial success, under certain conditions, they managed to achieve some of their objectives. Their major limitation though was their excessive cost. The main point of this investigation was to show that the demand for higher system throughput in Database Management Systems (DBMSs) can be achieved by combining off-the-self systems running on multiple but single-CPU hardware. Such generic hardware solutions are less costly and a lot more flexible.

Until recently, high throughput database processing was undertaken by a large mainframe that was the dedicated machine for all the data processing. During the last decade we have experienced a number of developments that are changing the way DBMSs are build and operate. First, we have seen the spectacular introduction and wide use of high--end workstations with very low prices. Second, disk units become larger and more reliable and finally, computer network technology has matured and offers reliable operations for file transfers, remote access and message handling. In this paper, we are concerned with the performance of modern Client-Server (CS) database architectures.

All CS architectures we have studied consist of a number of workstations (clients), one or more large workstation(s) or mainframe(s) which undertake the role of the server(s), and a local area network connecting them all together. We assume that multiple databases running on different servers are autonomous and that no inter-database transactions exist (which may cause inter-database blocking and additional delays). The client functionality ranges from just running the application with no caching on either main memory or disk and minimal or no decoupling between the client and the server, all the way to having full cache management capability on the client and high degree of decoupling and data distribution. Each server can have either a single or an array of disks for parallel I/O.

1.2. Evaluation of Client-Server Database Architectures

Several CS database architectures can be built with the above functionality. This investigation concentrated on the following: (a) no caching on the clients and single disk on each server. This is the Standard and minimal functionality configuration (SCS). (b) No caching on the clients but multiple disks on each server (CS-MD) for data replication and parallel I/O. (c) Enhanced disk cache management functionality on the clients for dynamic data migration and incremental maintenance of cached data, and a single disk on the server (ECS). Downloaded and cached data provides the replication needed for parallel I/O. (d) As in (c) with the addition of a special purpose buffer manager dedicated for facilitating the access of the server logs for the incremental cache management (ECS-LB).

We chose for evaluation the above (b-d) configurations as they are most comparable with regard to hardware (equal number of disks and roughly equivalent replication overhead). The SCS is used as the basis for our metric.

There are a number of studies that deal with similar issues to those we are discussing here. Hagman and Ferrari are among the first who tried to split the functionality of a database system and off--load parts of it to dedicated back--end machines. Among other results, they found that generally there is a 60 overhead in disk I/O and a suspected overhead of similar size for CPU cycles. Roussopoulos and Kang proposed the coupling of a number of workstation-based DBMSs loosely-coupled with a mainframe DBMS. The cooperation included incremental and lazy maintenance of dowloaded and cached data on the client databases.

We started our investigation in the area of database system architectures for heterogeneous information systems, the overhead of heterogeneous computing and the use of buffering and incremental computation models for alleviating the overhead. We then extended the Client-Server architectures to take advantage of the disks available on the clients to provide parallel access to replicated data.

In order to evaluate the benefits of such architectures, we have designed and implemented simulation packages for comparing several a tightly coupled architectures such as ordinary client-server, and loosely coupled ones such as the Enhanced Client-Server (ECS) and completely loose multidatabase architectures. Three such architectures are being studied and our first results are reported in [Roussopoulos & Delis 1991] and [Delis & Roussopoulos 1992].

The ECS architecture is based on *incremental computation models* [Roussopoulos 1991]. The results of queries originating from a workstation are materialized and incrementally maintained on the workstation. The architecture provides distributed functionality while ensuring high availability and low communication overhead. Incremental computation models combined with advanced indexing and storage strategies are used to achieve high performance [Roussopoulos et al 1993]. Gateways to multiple relational databases and to a network database are currently under development. Our implementation of the ADMS± prototype at the moment provides incremental access to Oracle, Ingres and ADMS. The prototype is used to reassure us about the results of the simulations.

The simulation was extended to measure the performance of all client-server architectures under variant sizes of main memory for buffering. The buffering clearly had an effect on the execution but did not change any of the trends observed in our non-buffering simulations.

We then extended our investigation in the area of database system architectures for heterogeneous and distributed information systems, the use of buffering and incremental computation models for alleviating the overhead of distributing computing [Delis & Roussopoulos 1992] and the benefits obtained by the parallel disk access observed in the ECS architecture. We were particularly interested in studying scalability of I/O systems and developed a prototype for measuring context switching and network latency overhead.

One of the biggest concerns in distributed database architectures is the management and the update control of dispersed and replicated data. The severe overhead in maintaining consistency resulted in the limited use of distributed systems or to systems that allow very restricted mode of updates. In the Enhanced Client-Server architecture, controlling the downloaded data is very important because this determines the performance. We apply incremental update algorithms to bring into sync the client data. However, if the client is disconnected for a long period, incremental may not be the most efficient way and retransmission of the whole data subset be more appropriate. However, to avoid this situation, a background process running in the client can trigger the incremental refresh. Another technique that can be used, is periodic propagation of updates to the clients. These techniques and alternative ones are examined in [Delis & Roussopoulos 1994].

2. Adaptive Database Systems

In the context of Adaptive DBMSs, we have concentrated on buffer management techniques, the design of dynamic query optimization techniques, and the concept of query feedback for adapting buffer allocation strategies.

2.1. Adaptive Buffer Management

Database buffers are used in most systems in order to avoid excessive disk IOs. Previous research on database buffer management primarily based on the classifications of page reference patterns exhibited by simple relational operations. However, fine classification and pattern prediction are difficult for complicated queries or in advanced data models.

Previous works on buffer allocation are based either exclusively on the availability of buffers at runtime or on the access patterns of queries. Our group introduced in [Ng et al 1991] a unified approach for buffer allocation in which both of these considerations are taken into account. Our approach is based on the notion of *marginal gains* which specify the expected reduction on page faults in allocating extra buffers to a query. In the first half of this paper, we present mathematical models and formulas computing marginal gains for various access patterns exhibited by relational database queries. The paper proposes a class of buffer allocation algorithms based on marginal gains. Finally, the paper presents simulation results that show that our approach is promising and our algorithms achieve significant improvements on throughput, as compared with previous methods.

Although in the above paper we proposed the marginal gains-based policy that takes both run-time and access patterns of queries into account, and allows flexible (sub-optimal) buffer allocations to queries, this method, as well as all previously proposed, is static, that is it uses pre-calculated parameters to make the buffer allocation decisions. In [Faloutsos et al 1991], we extend that idea in order to design an *adaptable* buffer allocation algorithm, that automatically optimizes itself for changing query workloads. The basic concept used is that of *predicting* the effect an allocation may have to the throughput of the system, and making the allocation that maximizes the expected throughput.

However, all these methods are heavily dependent on the assumed pattern of access and any deviation from these can cause thrashing. To remedy this, we proposed an efficient scheme based on query feedback to achieve better buffer utilization. We devise a feedback mechanism, Faulting Characteristic Model (FCM), [Chen & Roussopoulos 1993], to associate each executed query with a few records that characterize its page access and faulting history. Based on FCM, an allocation scheme, termed Marginal Gain Ratio (MGR), is employed to adjust the allocations for recurring queries. Another scheme, called Never-Used-Again (NUA), is used to optimize the buffer replacements. Experiment results show that, MGR and NUA outperform the classification-based methods by 15-20% and 20-70% in average system throughput, respectively.

2.2. Estimation of Selectivities Using the Query Feedback

We also developed a novel approach for estimating the record selectivities of database queries. The actual attribute value distribution is approximated by a curve-fitting function using a query feedback. This approach has the advantages of incurring no additional overhead for database statistics collection, and adapting to the evolving data value distributions during updates.

In most database systems, query optimization chooses efficient execution plans for database queries. In order to recognize the best plan, the optimizer must make accurate estimates of the costs of alternative plans. One of the most important factors that affect the plan cost is data selectivity, which is the number of records satisfying a given predicate. Accurate selectivity estimation attracted a great deal of research which resulted in a variety of methods. In summary, all these methods rely on the maintenance of the attribute value distributions obtained by having to do extra access to the database. In this research we are exploring a different approach for approximating the attribute distribution using query feedback. The idea is to use the size of the query results as feedback to re-shape (regress) the attribute distribution.

This approach has the following characteristics:

(1) *No additional overhead:* Since query feedback is obtained as a bonus of query execution, it eliminates additional dictionary accesses for maintaining statistics. This benefits even more systems in which database access is expensive (e.g. very large/tertiary, or mobile databases).

(2) *Statistics Accuracy:* Assumptions or estimates of certain statistics are always used to assist in planning an efficient query evaluation. Very often both the assumptions are wrong and statistics are inaccurate.. Query feedback provides high accuracy based on real experience,

and removes the need of having to do simplistic assumptions on value distributions such as uniformity or questionable rules of thumb.

(3) *System Independence:* Since query feedback, by itself, involves only the results rather than the internals of the query evaluations, it can be best customized for use in heterogeneous database systems. We are developing techniques which will introduce sample queries to run for the purpose of accurately estimating the cost of accessing heterogeneous DBMSs.

(4) *Efficiency:* Unlike the previous methods, no off-line database scan and statistical data gathering is needed to form the distribution. Such overhead overhead is proportional to the database size. In contrary, the overhead in the proposed method is constant time for each query, regardless of the database size.

This research was reported in [Chen & Roussopoulos 1994].

In a different direction, we developed a technique on *dynamic* or *parametric* query optimization. In existing query optimizers, the values of many important run-time parameters of the system, the data, or the query are unknown at query optimization time. Dynamic, or parametric, query optimization attempts to identify several execution plans, each one of which is optimal for a subset of all possible values of the run-time parameters. In [Ioannidis et al 1992] we present a general formulation of this problem and study it primarily for the buffer size parameter. We have adopted randomized algorithms as the main approach to this style of optimization and enhance them with a *sideways information passing* feature that increases their effectiveness in the new task. Experimental results of these enhanced algorithms show that they optimize queries for large numbers of buffer sizes in the same time needed by their conventional versions for a single buffer size, without much sacrifice in the output quality.

3. Conclusions

The major results of this project are in the areas of Incremental Access Methods, Client-Server Database architectures for heterogeneous DBMSs, adaptive buffer management techniques, and adaptive query optimization. In this final report a short description of these were outlined while the full details of these were published in the open literature.

During this three year contract, is period, we finished a prototype of an advanced Client-Server workstation-based DBMS which allows access to multiple heterogeneous commercial DBMSs. Experiments and simulations were then run to compare its performance with the standard client-server architectures.

Two new methods were discovered during this contract. The first is the notion of marginal gains to direct buffer allocation during query execution. The other was the concept of query feedback to make follow up query processing more efficient. This "learning from experience" idea was discovered somewhere in the middle of the contract and applied to several problems of query optimization. We also continued the fine-tuning of the incremental access methods in the client-server database architectures area along with a very extensive simulation on its scalability. It is worth pointing out, that a commercial DBMS vendor, Sybase, is now promoting one of the client-server architectures we proposed and doing incremental update processing as our ECS configuration.

The project has been very successful. It generated a total of fourteen papers. Six of them appeared in the top database journals (ACM Transactions of Database Systems, Transactions of IEEE on Knowledge and Data Engineering, Transactions on Software Engineering, and ACM Sigmod Record). Eight papers were presented at and published by the top international database conferences. Four of them appeared at the International Conference on Very Large Databases in

1991, 1992 and 1993. Two other papers appeared in the 1991 and 1994 ACM SIGMOD conferences. One in the 1994 International conference on Extending Database Technology, and one at the 1994 IEEE 14th International Conference on Distributed Computing Systems. The last paper received the Best Paper Award.

Nine students were partially supported by this contract. Three of them graduated with a PhD and three with a MSc. The other three are in various stages of their PhD program.

4. Recommendations

The research on advanced client-server database architectures is just finding its way to commercial applications. It will be beneficial to verify the results obtained by our simulations with real data from one of the many installations at NASA. This will test the scalability on a real network "in-action" as opposed to a simulation on a University local area network.

The research on adaptive database systems just put the foundation for this area. It has a lot of potential and can be exploited in many other systems areas where performance is critical. It has already generated a lot interest both to the University research community (follow up research is forthcoming from University of Wisconsin and University of Waterloo) and software industry (we got requests from Tandem, Sybase, and Oracle). This research should be continued not only for the database area but should be exported to the systems and architecture area. Several performance issues caused by overhead of interoperable distributed systems and network-based cooperative software will benefit from the concepts developed in this work. NASA should foster research in this area which will help in the replacement of todays monolithic software to a flexible component-based distributed and cooperative software of the future.

5. References

[Chen & Roussopoulos 1993]
Chen, C., Roussopoulos, N., "Adaptive Database Buffer Management Using Query Feedback," *Proceedings of the 1992 International Conference on Very Large Data* Bases, Dublin, Ireland, August 1993.

[Chen & Roussopoulos 1994]
C. Chen and N. Roussopoulos, "The Query Optimizer of ADMS," *The Fourth International Conference on Extending Database Technology*, St. John's College, Cambridge, UK, March 28-31, 1994.

[Chen & Roussopoulos 1994a]
C. Chen and N. Roussopoulos, "Adaptive Estimation of Selectivity Using Query Feedback," *ACM-SIGMOD International Conference on Management of Data*, Minneapolis, Minnesota, May 24-27, 1994. (also available as Tech. Report CS-TR-3197, University of Maryland, College Park).

[Delis & Roussopoulos 1993]
A. Delis and N. Roussopoulos, "Performance Comparison of Three Modern DBMS Architectures," *IEEE Transactions on Software Engineering*, Vol. 19, No, 2, February 1993.

[Roussopoulos et al 1991]
Roussopoulos, N., Mark, L., Sellis, T. Faloutsos, C., "An Architecture for High Performance Engineering Information Systems," *IEEE Trans. on Software Engineering*, Vol. 17, No. 1, pp. 22-33, January 1991.

[Delis & Roussopoulos 1993]
A. Delis and N. Roussopoulos, "Performance Comparison of Three Modern DBMS Architectures," *IEEE Transactions on Software Engineering*, Vol. 19, No, 2, February 1993.

[Delis & Roussopoulos 1992]
Delis, A. and Roussopoulos, N., "Performance and Scalability of Client-Server Database Architectures", Proceedings of the 18-th *International Conference on Very Large Data Bases*, Vancouver, August 23-27, 1992.

[Delis & Roussopoulos 1994]
A. Delis and N. Roussopoulos, "Management of Updates in the Enhanced Client-Server DBMS" Best Paper Award, *IEEE/CS 14th International Conference on Distributed Computing System s*, Poznan, Poland, June 1994.

[Faloutsos et al 1991]
Faloutsos, C., Ng, R., and Sellis, T. "Predictive Load Control for Flexible Buffer Allocation", Proceedings of the *1991 International Conference on Very Large Data Bases*, Barcelona, Spain,

September 1991.

[Ioannidis et al 1992]
Ioannidis, Y., Ng, R., Shim, K., and Sellis, T. ''Parametric Query Optimization'', Proceedings of the *1992 International Conference on Very Large Data Bases*, Vancouver, Canada, August 1992.

[Ng et al 1991]
Ng, R., Faloutsos, C., and Sellis, T. ''Flexible Buffer Allocation based on Marginal Gains'', *Proceedings of the 1991 ACM-SIGMOD International Conference on the Management of Data*, Denver, CO, May 1991.

[Roussopoulos 1991]
Roussopoulos, N., "The Incremental Access Method of ViewCache: Concept and Cost Analysis", *ACM Transactions on Database Systems*, September 1991.

[Roussopoulos & Delis 1991]
Roussopoulos N. and Delis, A., "Modern Client-Server DBMS Architectures," *SIGMOD Record,* Vol. 20, Mo 3, September 1991.

[Roussopoulos et al 1993]
Roussopoulos, N., Economou, N., Stamenas, A. "ADMS: A Testbed for Incremental Access Methods," *IEEEE Trans. on Knowledge and Data Engineering,* Vol. 5, No. 5, October 1993, pp. 762-774.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** August 31, 1994	**3. REPORT TYPE AND DATES COVERED** Contractor Report

4. TITLE AND SUBTITLE VIEWCACHE: An Incremental Database Access Method for Autonomous Interoperable Databases	**5. FUNDING NUMBERS** 930
6. AUTHOR(S) Nicholas Roussopoulos, Timos Sellis	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Maryland, Office of Research and Administration, 2100 Lee Building, College Park, MD 20742	**8. PERFORMING ORGANIZATION REPORT NUMBER** NAS5-32337 555-09
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration - HQ/ OSSA, Washington, D.C. 20546-0001; Universities Space Research Association, 10227 Wincopin Circle, Suite 212, Columbia, MD 21044	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER** CR-189384

11. SUPPLEMENTARY NOTES

Technical Monitor: J. Hollis, Code 930

12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified–Unlimited Subject Category 82 Report is available from the NASA Center for AeroSpace Information, 800 Elkridge Landing Road, Linthicum Heights, MD 21090; (301) 621-0390.	**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

It this research, we investigated a number of design and performance issues of interoperable database management systems. The major results of our investigation were obtained in the areas of Client-Server Database architectures for heterogeneous DBMSs, incremental computation models, buffer management techniques, and query optimization. We finished a prototype of an advanced Client-Server workstation-based DBMS which allows access to multiple heterogenous commercial DBMSs. Experiments and simulations were then run to compare its performance with the standard client-server architectures.

The focus of this research was on adaptive optimization methods of heterogeneous database systems. Adaptive buffer management accounts for the random and object-oriented access methods for which no known characterization of the access patterns exists. Adaptive query optimization means that value distributions and selectivities, which play the most significant role in query plan evaluation, are continuously refined to reflect the actual values as opposed to static one that are computed off-line.

Query feedback is a concept that was first introduced to the literature by our group. We employed query feedback for both adaptive buffer management and for computing value distributions and selectivities. For adaptive buffer management, we use the page faults of prior executions to achieve more "informed" management decisions. For the estimation of the distributions of the selectivities, we use curve-fitting techniques, such as "least squares" and "splines," for regressing on these values.

14. SUBJECT TERMS Client-Server Architectures, query optimization, buffer management, query feedback, adaptive database systems	**15. NUMBER OF PAGES** 16
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	Unlimited